Interprozeß-Kommunikation auf verteilten Systemen

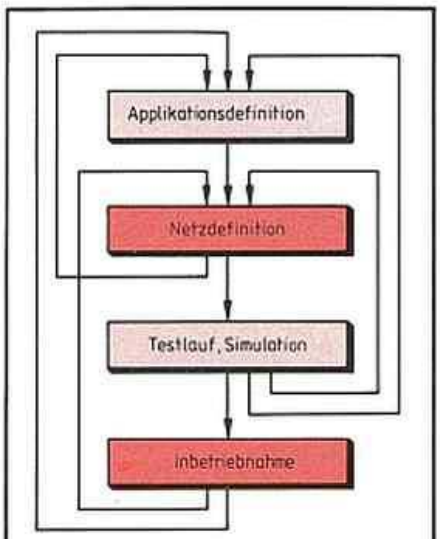
C. Eggers und A. Strabel beschreiben das IPK-System

Cord Eggers und Andreas Strabel sind bei der ITRH beschäftigt. Das beschriebene Softwarepaket entstand im Rahmen des Projektes „VME in der VAX", ein in einen VAX-Rechner integriertes VMEbus-Subsystem.

Das Softwarepaket „Interprozeßkommunikation" (Arbeitsname IPK-System) ermöglicht die Entwicklung von parallel ablaufenden Anwendungen, die in hohem Maße systemunabhängig sind. Das kommt insbesondere Entwicklungen auf unterschiedlichen Rechnern mit verschiedenen Betriebssystemen (z. B. VAX/VME und VMEbus/OS-9) zugute. Denn häufig steht der Entwickler vor dem Problem, das komfortable Betriebssystem VMS mit der schnellen Interrupt-Verarbeitung von VMEbus-Systemen zu kombinieren.

„Virtuelle Rechner"

Auf der einen Seite ist die DEC-Welt mit ihrem komfortablen und bewährten Betriebssystem VMS, dessen Unterstützung und Weiterentwicklung auf lange Zeit gewährleistet ist. Der hohe Komfort des Betriebssystems hat aber – insbesondere bei überwiegend technischen Applikationen, die trotz der Bemühungen von DEC, im kommerziellen Sektor Fuß zu fassen, immer noch das Haupteinsatzgebiet der DEC-Rechner darstellen – einen gravierenden Nachteil zur Folge: aufgrund der Eigenschaften des Q-Bus und der internen Struktur des VMS sind Interruptbehandlungen im Nanosekunden- bzw. unteren Mikrosekundenbereich nicht möglich. Einigermaßen anspruchsvolle Echtzeit-Anwendungen können daher heutzutage von einer VAX mit VMS allein nicht mehr bewältigt werden.



Ablauf eines Applikationsentwurfs: Nach dem Test bzw. der Simulation kann es erforderlich sein, die Anwendung oder das Prozessornetz umzudefinieren

Dem steht auf der anderen Seite die VMEbus-Welt mit einer großen Anzahl von Baugruppen gegenüber. Sie sind in der Lage, Echtzeit-Anwendungen mit Reaktionszeiten im Nanosekundenbereich gerecht zu werden. Modularität und Flexibilität sind weitere Eigenschaften der VMEbus-Systeme. Einige Posten allerdings sind auch auf der Soll-Seite zu finden: geringe Software-Unterstützung, Betriebssysteme, die hinsichtlich Komfort und Reifegrad keinen Vergleich mit einem so etablierten Betriebssystem wie VMS standhalten und schließlich die so gut wie nicht vorhandene Anwendersoftware.

Hinzuzufügen ist allerdings, daß inzwischen einige einigermaßen bekannte (und anerkannte) Betriebssysteme, etwa OS-9 oder Versados, vorhanden sind. Diese erscheinen zwar, was Umfang und Komfort angeht, geradezu ärmlich im Vergleich zu VMS, andererseits warten sie aber mit Eigenschaften auf, die einen Einsatz in Echtzeit-Anwendungen möglich machen – das haben sie eben VMS voraus.

Zwischen beiden Welten steht nun der Entwickler, der sich vor die Aufgabe gestellt sieht, eine Applikation zu entwerfen, die nur dadurch angemessen unterstützt werden kann, daß die besten Eigenschaften beider Welten zusammengebracht und in einem System vereint werden. Eine Applikation etwa, die mehrere Ebenen mit ihren oft völlig unterschiedlichen Anforderungen zu bedienen hat, wie z. B. die Prozeßsteuerungsebene mit der Forderung nach Echtzeit-Fähigkeit oder die höheren Ebenen, wie Abteilungs- und Unternehmensebene, mit ihren Forderungen nach einem komfortablen Datenverwaltungssystem. Dies alles ist typisch bei CIM- und ähnlichen Anwendungen.

Herkömmlicherweise wird ein Entwickler in dieser Situation nun daran gehen müssen, Hard- und Softwarekomponenten (Rechner,Betriebssysteme, spezielle Komponenten wie D/A- und A/D-Wandler) mit den zugehörigen Entwicklungs-Werkzeugen auszuwählen. Dann sind sie gegebenenfalls über ein ebenfalls zu bestimmendes Netz in ein Gesamtsystem zu integrieren. Unglücklicherweise hat das zu geschehen, bevor die an sich wichtigere Aufgabe der Abbildung einer Applikation auf verteilte Prozesse (unabhängig von jeder Hardware!) vorgenommen werden kann.

Diese frühe und enge Bindung von Teilen einer Applikation und der sie ausführenden Prozesse an spezielle Hard- und Software-Komponenten hat höhere Entwicklungskosten zur Folge. Das ist darauf zurückzuführen, daß die Entwicklungsoberfläche nicht einheitlich ist und die Wartung sowie Änderbarkeit sehr schwerfällig und mit hoher Hardware-Abhängigkeit verbunden ist.

Dieses Problem ist nicht neu. Einige Schlagworte zu seiner Überwindung sind:

Bestandteile des IPK-Systems

1. „IPK-Lib“: enthält Module mit Funktionen zur Verwaltung und Nutzung logischer Prozesse, Ereignisse, Kanäle und Semaphore.

2. „IPK-Manager“: Hintergrundprozeß auf jedem Knoten; zuständig für Netzverwaltung, Databank-Verwaltung, Kommunikationsdienste usw.

3. „IPK-Access“: Command-Line-Interpreter um interaktiv mit dem IPK-Manager zu arbeiten (Netzkonfiguration, Datenbank einrichten oder ändern, Statistik-Daten erfragen usw.).

4. „IPK-Tools“: Verschiedene Dienstleistungen, wie Dateitransfer, Backup usw.

- virtuelle Rechner
- verteilte Systeme
- verteilte Datenbanken

Auch das Ingenieurteam der IRTH in Hamburg sah sich vor dieses Problem gestellt. Ausgehend von den oben geschilderten Beobachtungen in der DEC- und VMEbus-Welt begann man, sich darüber Gedanken zu machen, wie sich die Vorteile der beiden Welten miteinander verbinden lassen, ohne deren jeweilige Nachteile in Kauf nehmen zu müssen.

Nachdem der Buskoppler (siehe VMEbus 1987, H. 4, S. 42 ff.) als das erste Ergebnis dieser Überlegungen, seine Feuertaufe überstanden und sich in praktischen Einsätzen bewährt hatte, wurde beschlossen, im Rahmen eines längerfristigen Projektes, die bis dahin schon vorhandenen, aber mehr oder weniger nebeneinanderlaufenden Ideen zur Nutzung dieses Kopplers (z. B. virtuelles Terminal, virtuelle Platte) in ein darüberstehendes Gesamtkonzept zu integrieren. Im Zentrum dieses Konzepts steht die Idee, Applikationen unabhängig von Hardware-Überlegungen entwickeln zu können, also so zu tun, als liefe die Applikation auf einem „virtuellen Rechner“ ab.

Realisierung einer Anwendung

In der Definitionsphase wird die Applikation in eine Reihe von parallel laufenden, um Betriebsmittel konkurrierende Prozesse, zerlegt. Es wird scheinbar keine Rücksicht darauf genommen, auf wievielen und welchen Rechnern später diese Prozesse laufen werden. Entscheidend für diesen Zerlegungsprozeß ist die Aufteilung der Applikation in klar unterscheidbare Teilaufgaben (z. B. Meßdatenerfassung, -verarbeitung, -komprimierung, -sicherung, -auswertung).

Um das Zusammenspiel der Prozesse – Kontrolle, Steuerung und Synchronisation von Prozessen, Verwaltung von Daten- und Kommunkationswegen – hardware-unabhängig zu halten, müssen in dieser Phase logische Namen für Prozesse, Ereignisse (Events), Semaphore und Kanäle benutzt werden. Das „Logische Namenskonzept“ wird weiter unten genauer beschrieben.

Programme, welche solche Prozesse darstellen, können bereits auf dieser Stufe geschrieben werden, am besten in einer portablen Hochsprache (z. B. „C“).

Bei der Definition eines Prozessornetzes wird die der Applikation zugrundeliegende „virtuelle Maschine“ auf die zur Verfügung stehende Hardware abgebildet. Aufgrund von zu erwartenden Rechen- und Kommunikationsanforderungen werden Rechner und Kommunikationseinheiten ausgewählt. Um die Rechner aufgrund der vorhandenen Kommunikationswege miteinander zu verbinden, wird ein entsprechendes Prozessornetz konfiguriert. Hierzu wird auf jedem Rechner ein spezieller Hintergrundprozeß (IPK-Manager) gestartet und mit den Konfigurationsdaten initialisiert.

Vor der Inbetriebnahme muß noch ein Testlauf durchgeführt werden. Je nach Typ der hier festgestellten Fehler oder Mängel, müssen Korrekturen entweder am Prozessornetz (Phase 2) oder gar an der Applikation selbst (Phase 1) vorgenommen werden.

Logisches Namenskonzept

Das Konzept der logischen Namen wurde eingeführt, um Hardware- und Betriebssystem-Unabhängigkeit zu erreichen. Erst zur Laufzeit wird eine Verknüpfung von logischem mit physischem Namen hergestellt. Eine solche Verbindung kann auch interaktiv definiert werden.

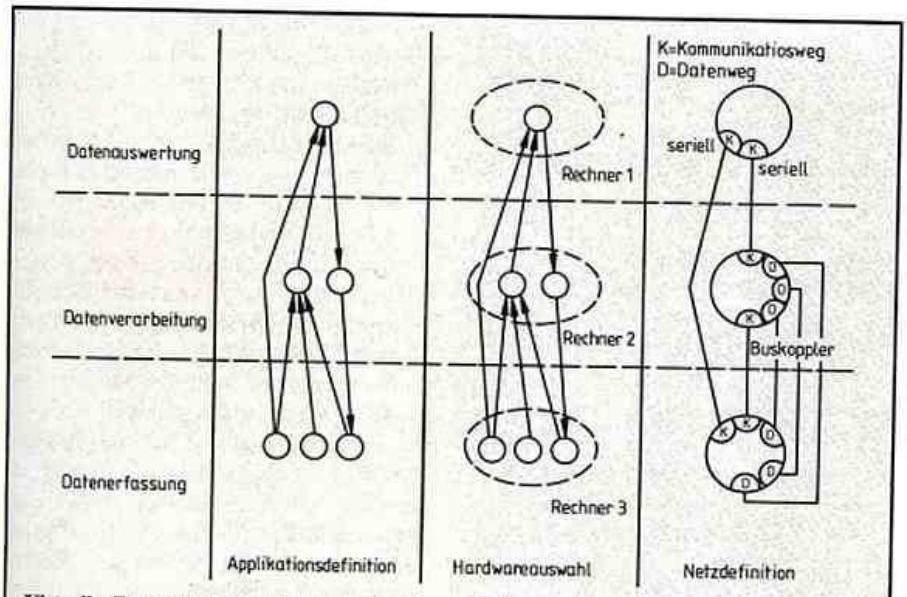
Alle logischen Namen können lokal oder global sein. Ein lokaler logischer Name ist für immer mit einem logischen Knotennamen verbunden und nur auf dem zugehörigen Knoten bekannt. Globale logische Namen dagegen sind der ganzen Applikation bekannt.

Folgende Typen von logischen Namen existieren:

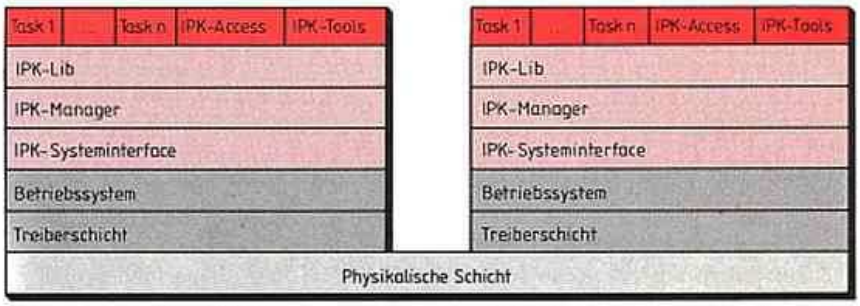
- node
- process
- event
- channel
- semaphore

Für alle Typen gelten folgende Grundoperationen:

- create – definiert einen logischen Namen und seine Übersetzung ▷



Virtuelle Rechner: Beim Entwurf mit dem IPK-System wird die Anwendung auf die virtuellen Rechner verteilt und dann definiert man die physikalischen Verbindungen



Zusammenspiel der IPK-Elemente: Die einzelnen Module des IPK-Systems können dem ISO/OSI-Schichtenmodell zugeordnet werden

- delete – löscht einen definierten logischen Namen
- link – stellt Verbindung zwischen Prozeß und logischem Namen her
- unlink – hebt diese Verbindung auf
- read – gibt Zustand zurück

Zusätzlich existieren typ-spezifische Operationen, von denen hier einige aufgelistet werden.

Für den „process"-Typ:
- run – Start eines Prozesses, der durch den logischen Namen identifiziert wird
- send–message – sende Meldung an einen Prozeß
- read–message – empfange Meldung eines Prozesses

Für den „event"-Typ:
- set – Setzen eines Events auf wahr (eingetreten) oder falsch (nicht eingetreten)
- get – Zustand eines Events einlesen
- wait – Warten auf bestimmten Zustand eines Events

Für den „channel"-Typ:
- read–ch – Empfangen der Datenblöcke eines anderen Prozesses, vorwiegend für große Datenmengen gedacht, die schnell übertragen werden müssen (im Gegensatz zu read–message kaum Software-Overhead. Entspricht Verbindung auf Schicht 2 nach ISO/OSI)
- write–ch – wie read–ch, nur andere Richtung.

Für den „semaphore"-Typ:
- lock – Ressource exclusiv belegen
- unlock – Ressource freigeben

Konzept des Prozessorennetzes

Um den bei der Planung einer Applikation in der Definitionsphase entstehenden „virtuellen Rechner" auf vorhandene Hardware abzubilden, ist eine Software vernetzter Prozesse entwickelt worden. Die damit zu realisierenden Netze sind durch folgende Eigenschaften charakterisiert:
- Unterstützung dezentraler Maschennetze;
- Kommunikationskanäle (exclusiv für die IPK-Software);
- Datenkanäle für direkte Verbindung zweier Prozesse (schneller Datenaustausch);
- Routing über halbdynamische Routing-Tabellen;

Die eigentliche Netzsoftware besteht aus der IPK-Datenbasis (Tabellen logischer Namen, Routing-Tabellen und Link-Tabellen), dem IPK-Manager (Prozess-, Event-, Channel- und Semaphore-Manager), dem IPK-Kern (Netz-, Queue-, Time-, Security- und Error-Manager) sowie der Schnittstelle zum Betriebssystem.

Kommunikationskonzept

Das IPK-System unterstützt zwei Kommunikationsarten:
- Austausch von relativ kurzen Nachrichten, meistens in Textform und in größeren zeitlichen Abständen (Terminal-Ein-/Ausgabe). Dieser Nachrichtenaustausch erfolgt unter Kontrolle des IPK-Managers.
- Schnelle Übertragung von großen Datenmengen, fast immer in Binärformat. Der IPK-Manager ist hier fast gänzlich ausgeschaltet und wird nur zur Bereitstellung eines Kanals benötigt. Der eigentliche Transfer wird von den beteiligten Prozessen selbst gesteuert. Ist der Datentransfer beendet, wird der Kanal dem IPKSystem zurückgegeben.

IPK-System unterstützt zunächst den von IRTH entwickelten Buskoppler, da er eine besonders schnelle DMA-Einheit darstellt (praktische Transferrate ungefähr 700 kBit/s) und für kleinere Datenströme serielle Schnittstellen. Prinzipiell sind aber auch weitere Kommunikationswege möglich. □

VORSCHAU

Ing.-Büros, Software- und Systemhäuser: Fragebogen anfordern!

In Ausgabe 5 des Fachmagazins „VMEbus" erscheint eine Übersicht über Ingenieurbüros, Software- und Systemhäuser, die im Auftrag von Kunden VMEbus-Systeme integrieren und programmieren, d. h., schlüsselfertige Lösungen liefern. Um eine möglichst umfassende Übersicht zu bieten, bittet die Redaktion darum, daß sich die Firmen, die noch keinen Fragebogen bekommen haben, aber in diese Übersicht aufgenommen werden wollen, sich umgehend bei der Redaktion einen Fragebogen anfordern. Einsendeschluß ist der 10. August 1988 (keinen Tag später).

Zusatzkarten für Sun-Workstation

Wer bietet Erweiterungskarten für die Sun-Workstations an, die mit einem VMEbus ausgerüstet sind? Die Redaktion des Magazins „VMEbus" plant für die Oktober-Ausgabe, Heft 5, eine entsprechende Marktübersicht. Außerdem soll in diesem Heft schwerpunktmäßig über VMEbus-Workstations berichtet werden. Die Redaktion sucht deshalb noch Autoren, die VMEbus-Workstations erweitert haben (z.B. um Bildverarbeitungs-Karten, Datenerfassungs-Platinen). Redaktionsanschrift: Holger Zeltwanger, Postfach 91 03 41, 8500 Nürnberg 91, Tel. (09 11) 33 59 08.